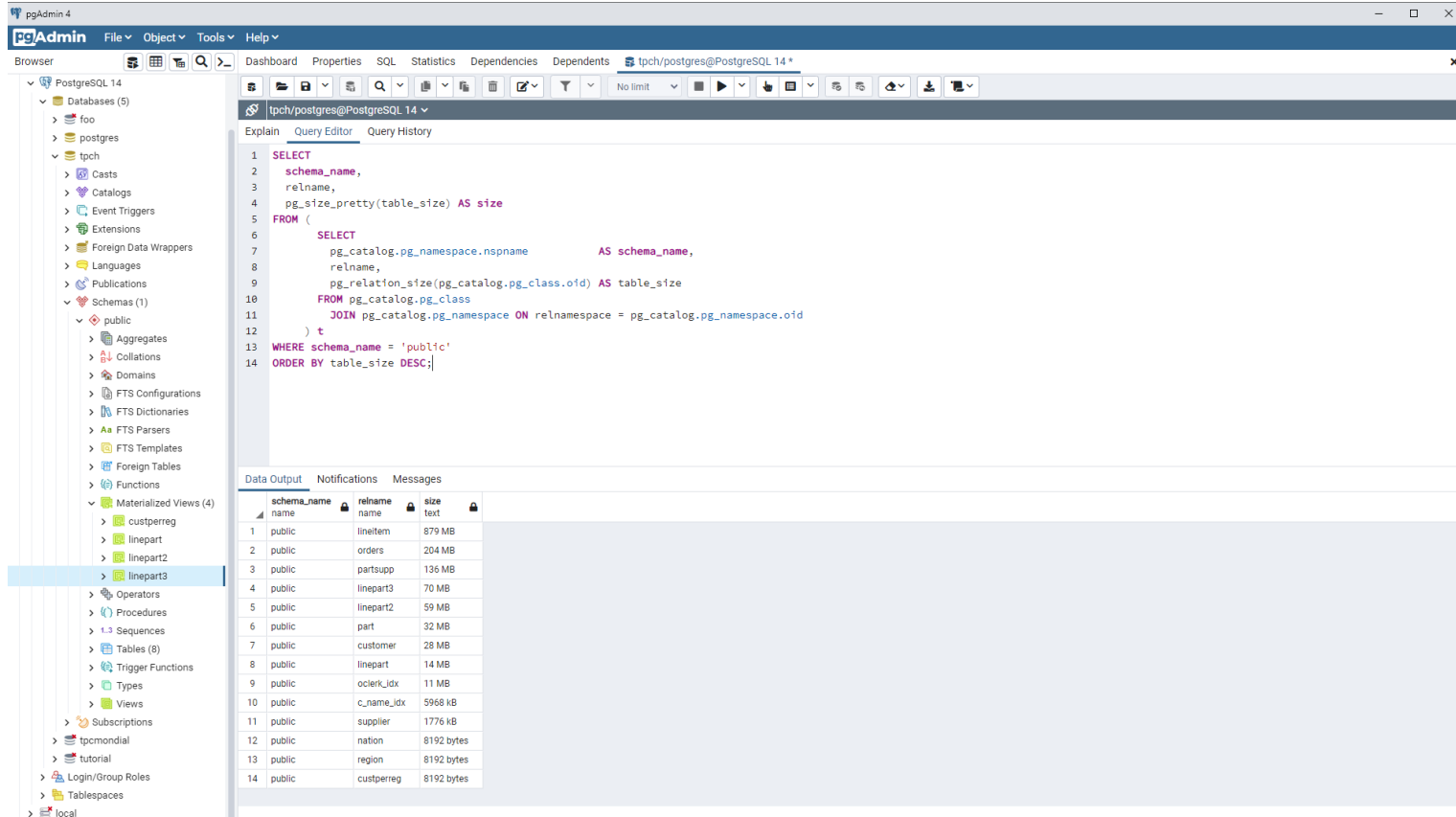


Schema Integration

Scalable Data Engineering

Addendum Exercise



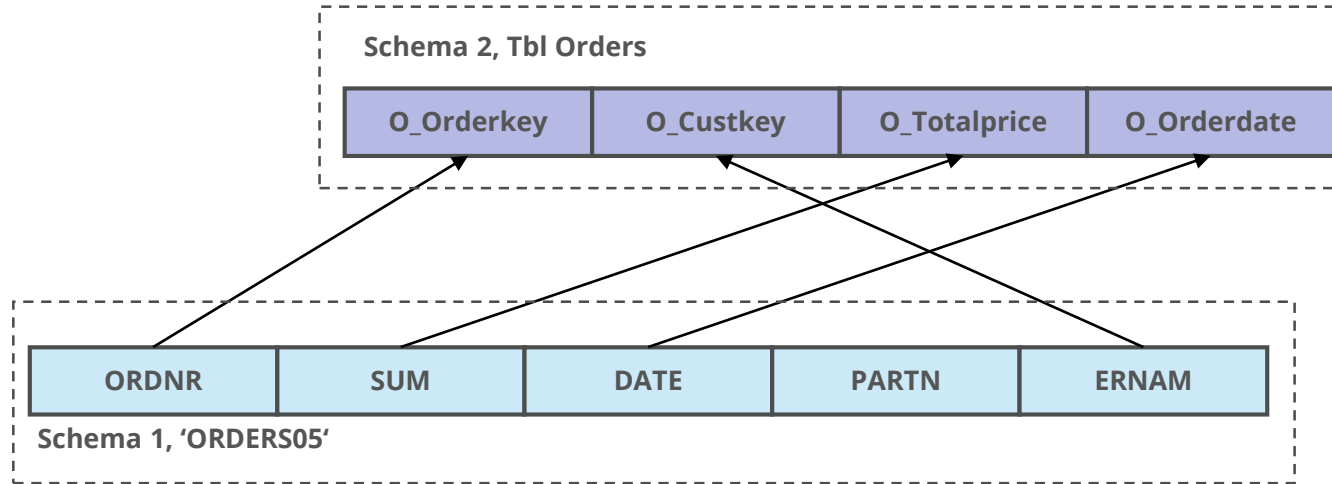
The screenshot shows the pgAdmin 4 web interface. On the left, the 'tpch' database is selected in the 'Databases (5)' tree. The 'Query Editor' tab is active, displaying a SQL query that lists tables in the 'public' schema, ordered by size in descending order. The query is as follows:

```
1 SELECT
2     schema_name,
3     relname,
4     pg_size_pretty(table_size) AS size
5 FROM (
6     SELECT
7         pg_catalog.pg_namespace.nspname           AS schema_name,
8         relname,
9         pg_relation_size(pg_catalog.pg_class.oid) AS table_size
10    FROM pg_catalog.pg_class
11   JOIN pg_catalog.pg_namespace ON relnamespace = pg_catalog.pg_namespace.oid
12 ) t
13 WHERE schema_name = 'public'
14 ORDER BY table_size DESC;
```

Below the query editor, the 'Data Output' tab shows the results of the query. The results are displayed in a table with the following columns: schema_name, relname, and size. The tables are listed in descending order of size.

schema_name	relname	size
public	lineitem	879 MB
public	orders	204 MB
public	partsupp	136 MB
public	linepart3	70 MB
public	linepart2	59 MB
public	part	32 MB
public	customer	28 MB
public	linepart	14 MB
public	oclerik_idx	11 MB
public	c_name_idx	5968 kB
public	supplier	1776 kB
public	nation	8192 bytes
public	region	8192 bytes
public	custperreg	8192 bytes

Problem

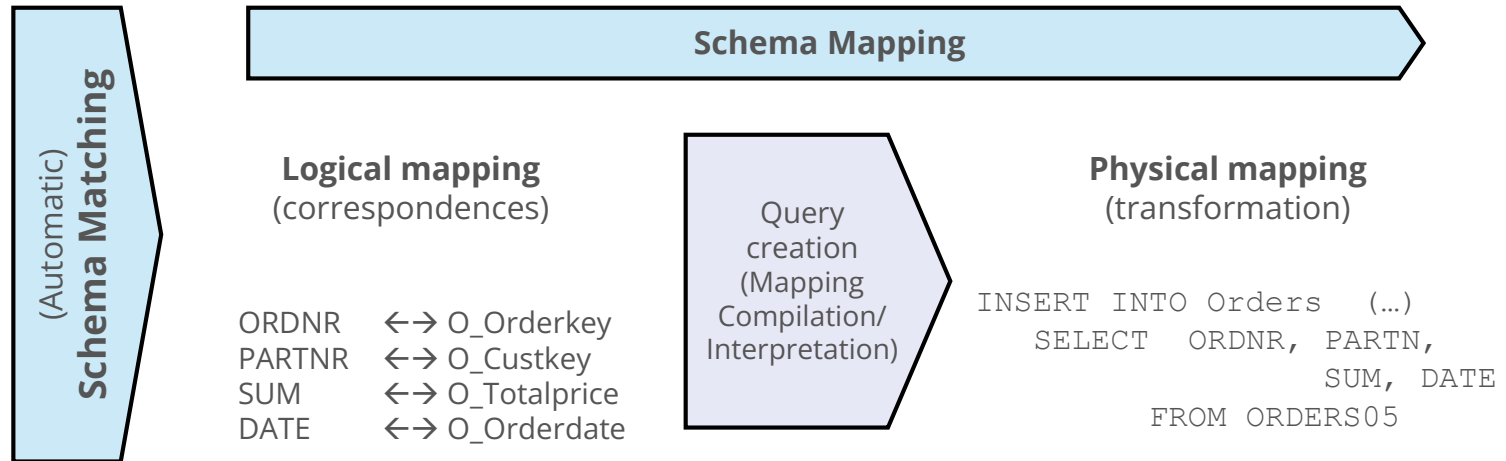


Example: SAP ERP R/3 (2007)

- **67.000** tables
- **700.000** columns
- **10.000** views
- **13.000** indexes

Terms

- Schema mapping: mapping one schema to another schema
- Schema matching: automatic identification of schema mappings



Motivation

1) Schema integration

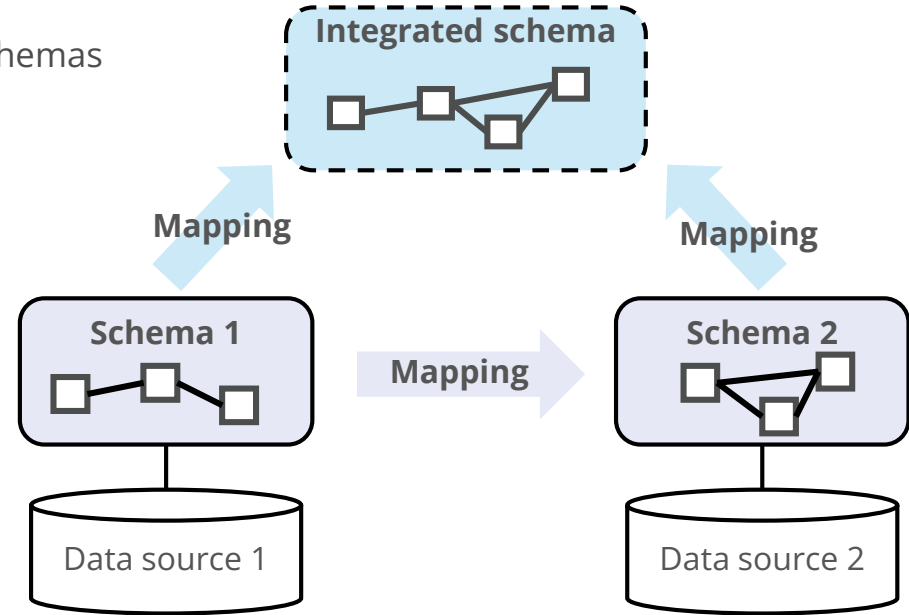
- Mapping existing schemas to one consolidated schema
- Target schema has no own semantic...
- ...it rather represents semantics of the source schemas

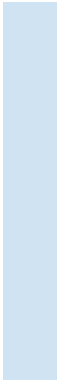
2) Schema Mapping

- Mapping existing schemas to each other

Challenge

- Both approaches must overcome semantic and structural heterogeneity





Schema Integration

Definition

- Aggregate multiple independent schemas into one global schema
- Elimination of overlaps and redundancies, ensure consistency of data via integration

Characteristics of a global (integrated) Schema

- Completeness
 - Items of all local schemas are reflected in the global schema, not always necessary
- Correctness
 - Semantics of local sources are kept (integrity constraints, cardinalities...), DWH consistency
- Minimality
 - Global schema is free of redundancy
- Understandability
 - All transformations and processes are documented

→ Cannot be tested algorithmically

Two-phase process

1. *Pre-integration phase*

- Analyze and document source schemas
- Starting schemas have bigger impact on the integration result
- Identify schema similarities and differences
- Determine the integration strategy that should be used (binary / n-ary integration strategies) → strategy implies schema preferences

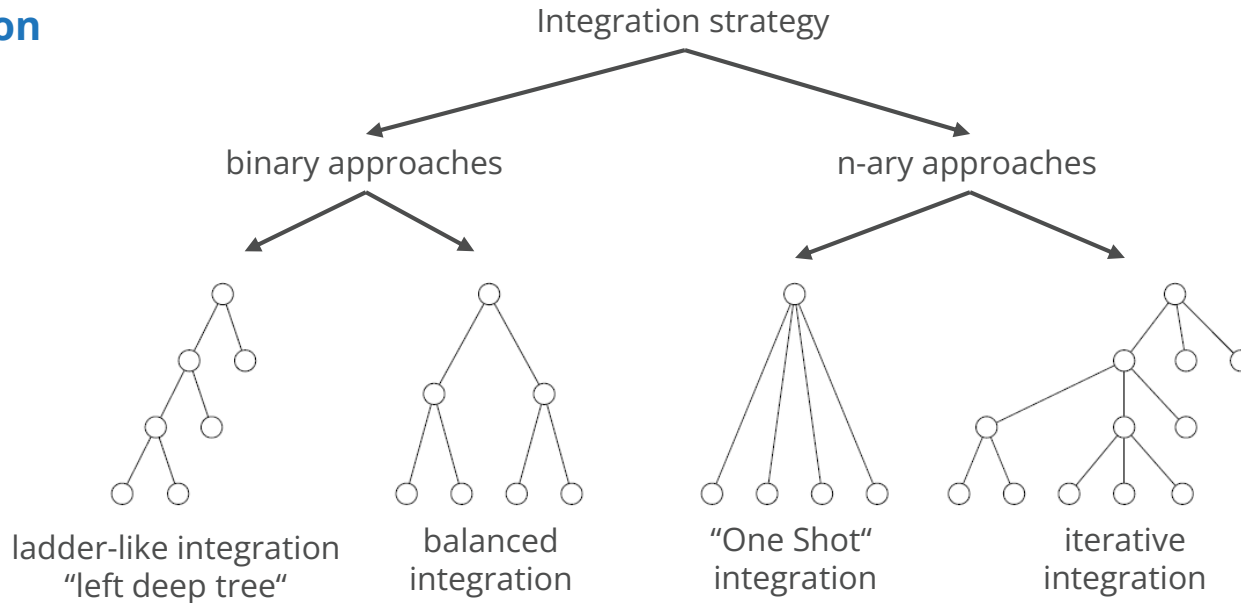
2. *Actual integration phase*

- Modify involved schemas

Integration strategies

- Course of action for the definition of the integrated schema

Classification

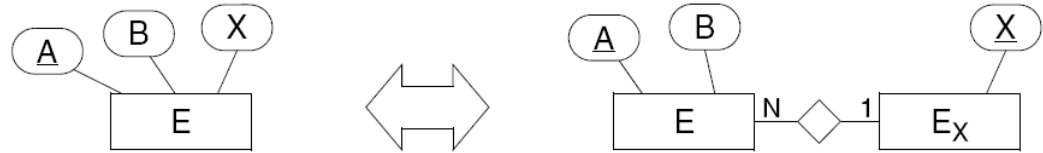


Problem: ambiguities in modelling

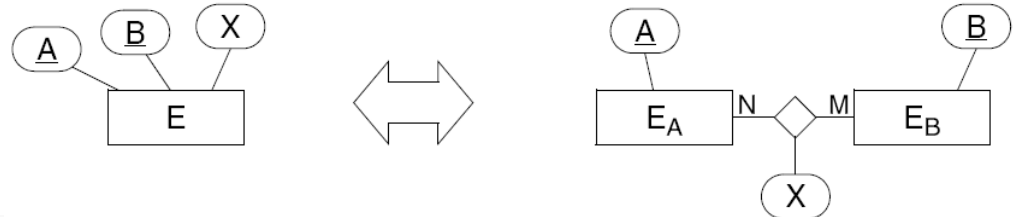
- Objects and relations are modelled differently
 - Domain-specific way of thinking, oriented towards different physical implementations
- Adjustment of different modelling necessary

Example: different adjustments

- Transform attributes and entities

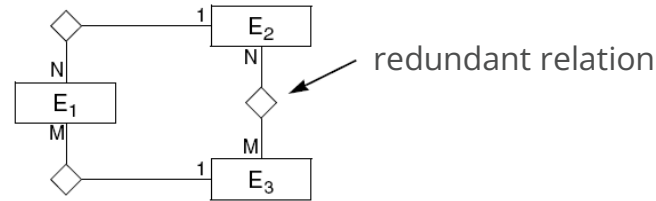


- Transform composite primary key (star schema)

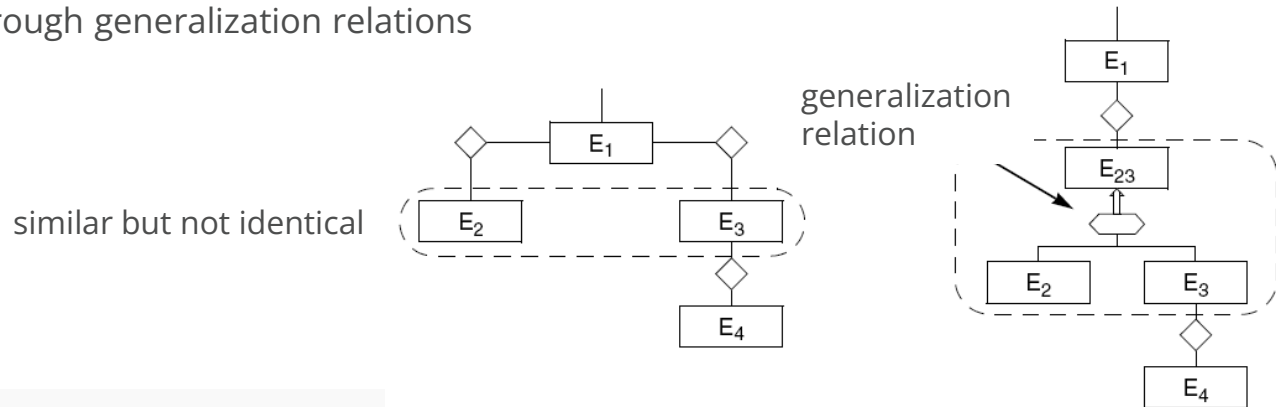


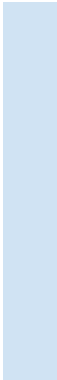
Example: redundancy elimination and simplification

- Eliminate redundancies (structural simplification)



- Simplification through generalization relations





Schema Mapping

(Inter-schema) correspondence

- Assignment of one/multiple elements of the source schema to one/multiple elements of the target schema

(High-level) mapping

- A set of correspondences between two schemas

(Low-level) logical mapping

- Logical translation of mappings, that fulfills the integrity constraints of both schemas

Interpretation

- Translation of a logical mapping into a transformation query

Transformation query

- Query (e.g. SQL/XQuery), that translates data of the source schema into structure of target schema

Motivation and Challenges

Transformation of data between heterogeneous schemas

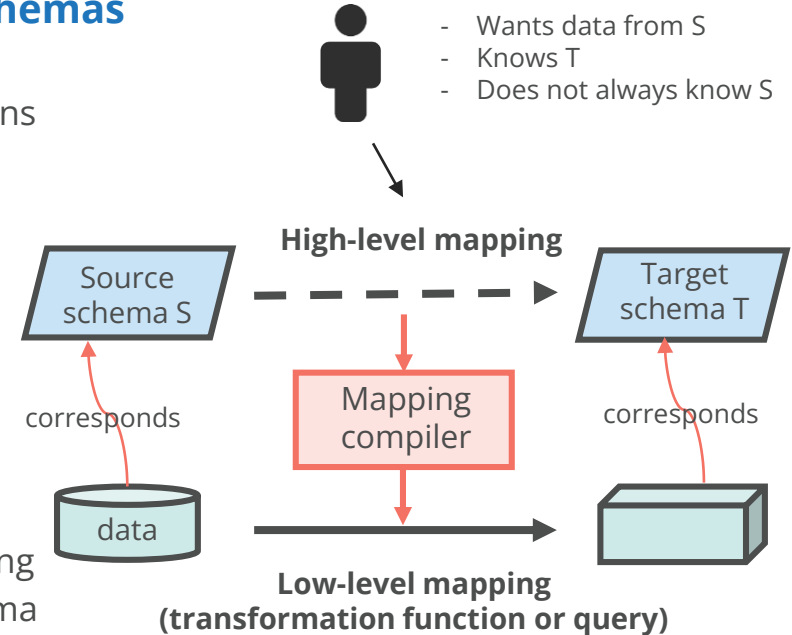
- Old but recurring problem
- Typically, domain experts write complex queries or functions
- XML increases complexity: XML schema, Xquery

Idea: automation

- **given:** two schemas and high-level mapping
- **wanted:** query for data transformation

Challenges

- Generate the “right” query considering schema and mapping
- Guarantee that transformed data satisfies the target schema



Motivation and Challenges

Support for nested structures

- Nested relational model
- XML
- Nested integrity constraints

Preservation of data semantics

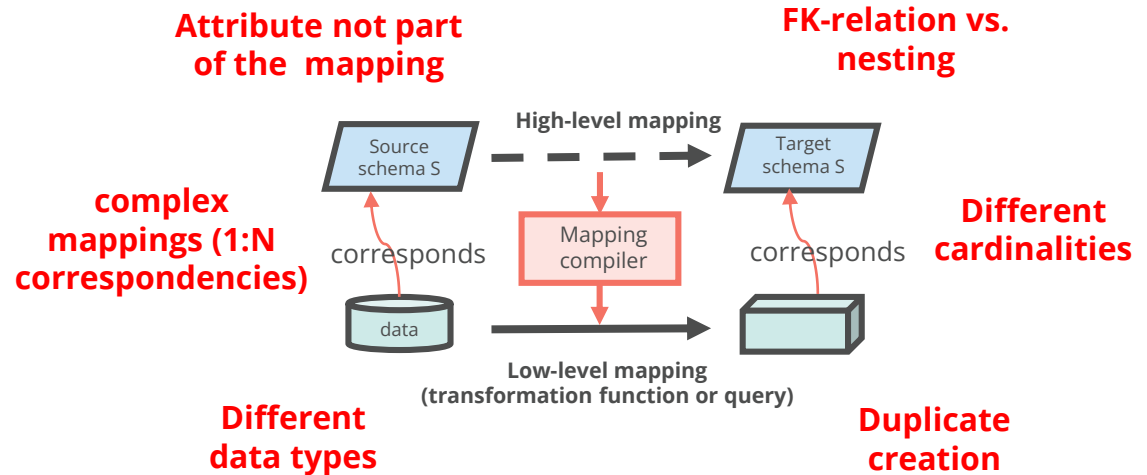
- Identification of associations
- Schemas and integrity constraints

Generation of new data

Recognize user intention

Efficient data transformation

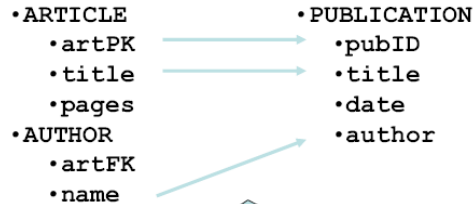
- For materialization (incremental execution), for virtual integration (query-unfolding)



Normalized vs. denormalized representation of 1:N-associations

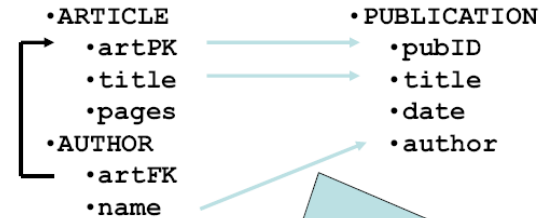
- Denormalization via intra-schema correspondence

Denormalization

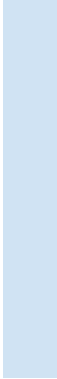


```
SELECT artPK AS pubID  UNION  SELECT null AS pubID
      title AS title    null AS title
      null AS date      null AS date
      null AS author    name AS author
FROM   ARTICLE          FROM   AUTHOR
```

Denormalization (2)

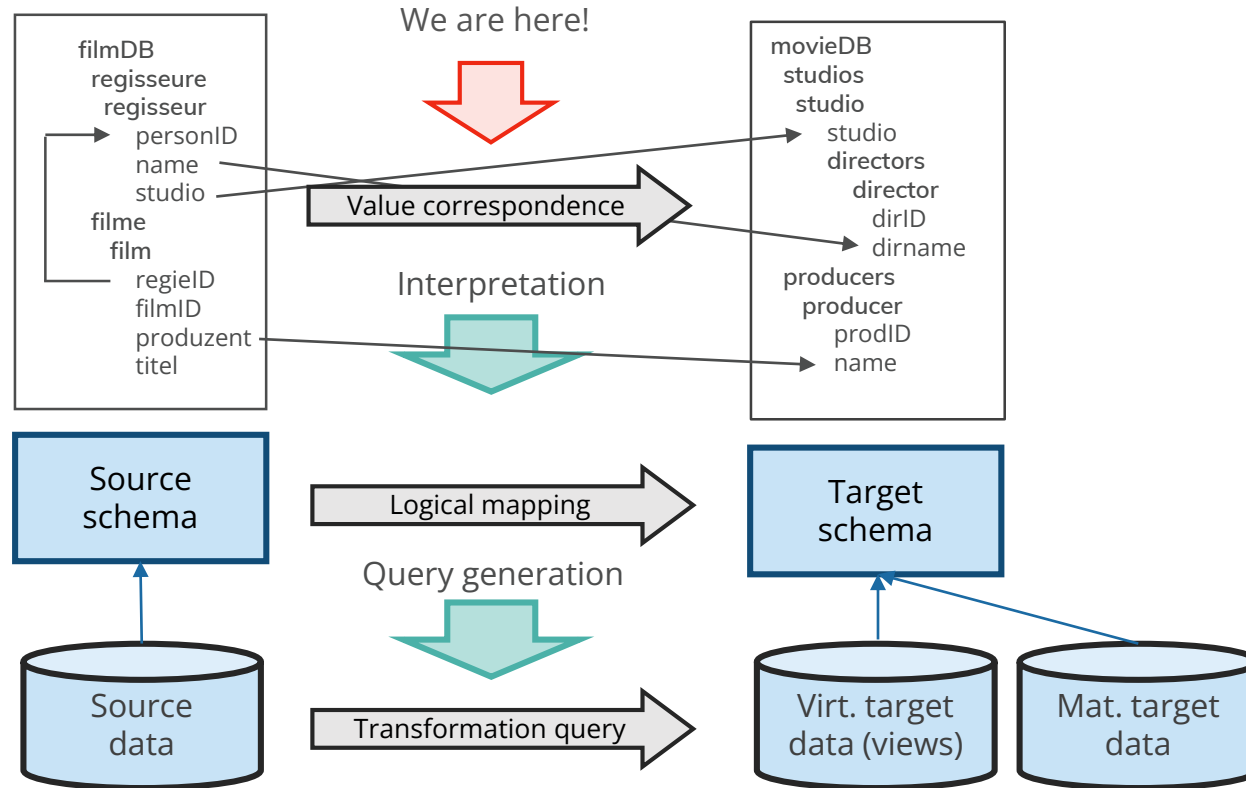


```
SELECT      artPK AS pubID
            title AS title
            null AS date
            name AS author
FROM        ARTICLE, AUTHOR
WHERE       ARTICLE.artPK = AUTHOR.artFK
```

Schema Matching

Motivation and Challenges



Challenges

- Big schemas
 - > 100 tables, many attributes
- Confusing schemas
 - Deep nesting, foreign keys
 - XML schema
- Heterogeneity
 - Synonyms, homonyms
 - Foreign-language schemas
 - Cryptic Schemas
 - Attribute names ≤ 8 characters
 - Table names ≤ 8 characters

Conclusion

- High complexity, $O(n \cdot m)$
- Cannot be done manually, at least parts have to be automated
- Semi-automatic matching reduces work by 75%

Definitions

Schema

- Set of elements, connected by a structure

Schema-matching

- Automatically generates correspondences
- Give: two schemas S1 and S2
 - Schema information, instance data
 - Support dictionaries
- Wanted: mapping
 - Correspondences between elements from S1 and S2
 - Similarity value in [0,1]

Quality

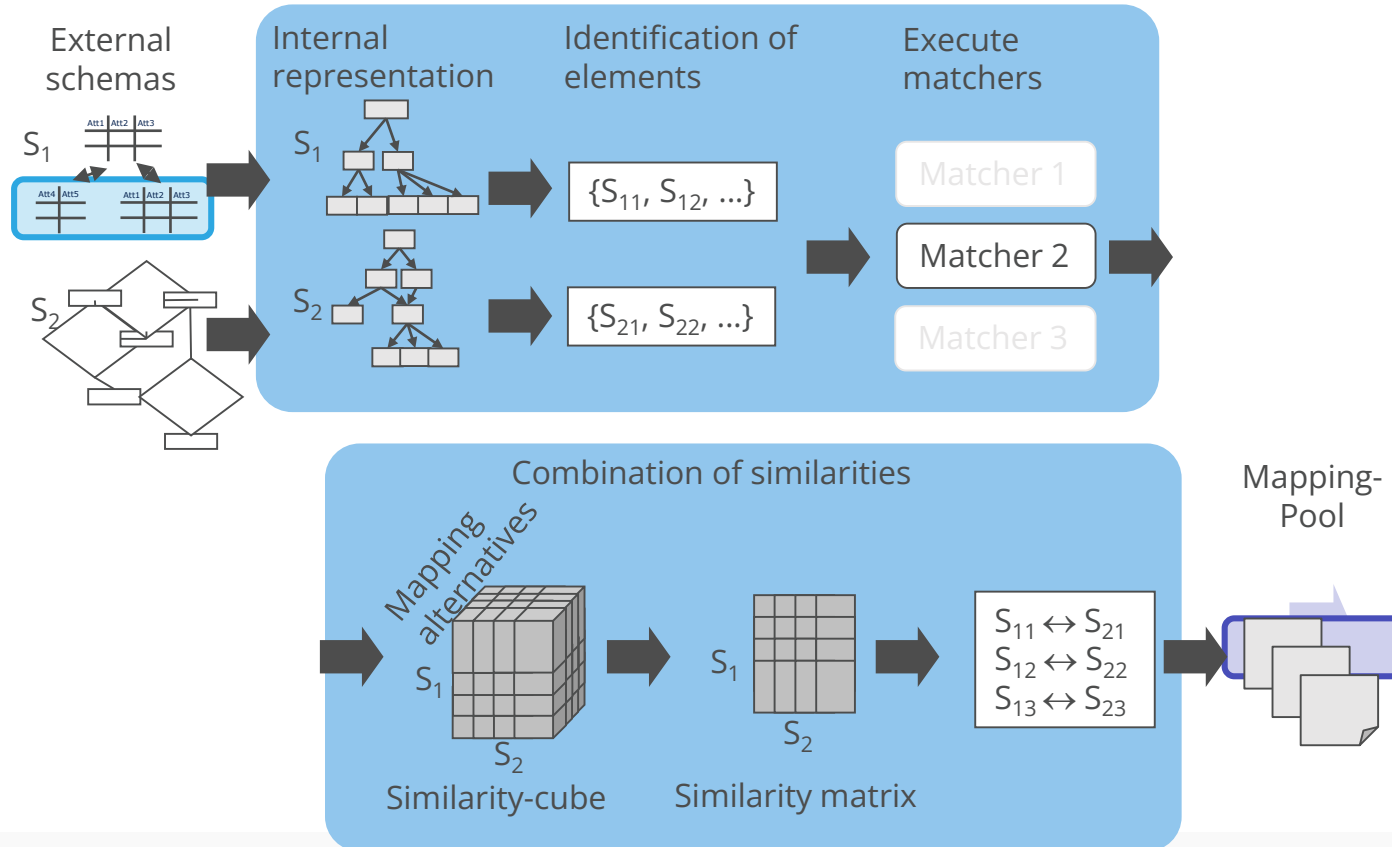
- Precision, Recall, F-Measure
- Benchmarks: XBenchMatch, STBenchmark

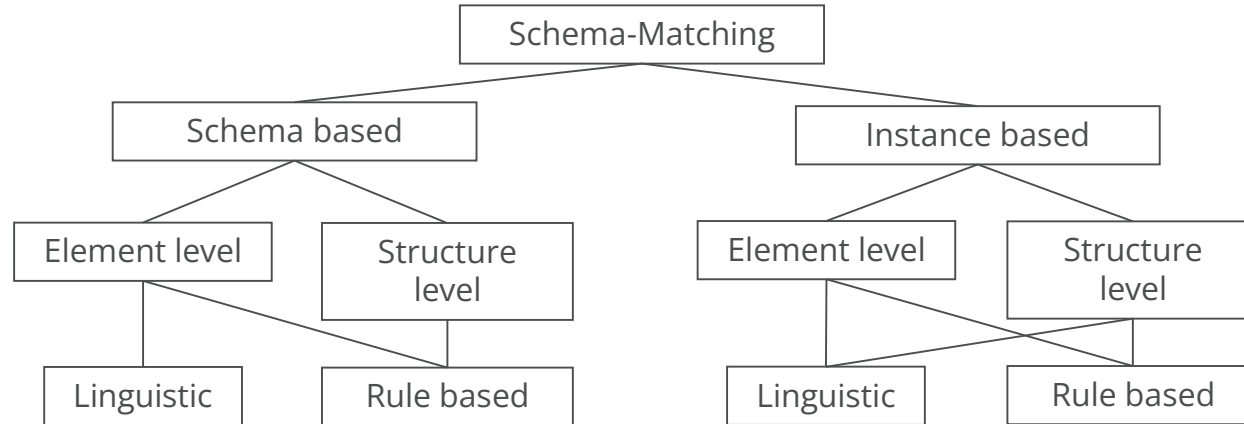
	C_Na	C_St	C_C
Customer	Emma Schmidt	Luisenstraße 90	München, 80333
	Klaus Schumann	Königstraße 7	Dresden, 01199
	...	...	...



Person	Firstname	Lastame	Address			
			Street	Nr.	Zip	City
	Susanne	Fröhlich	Weststraße	2	01187	Dresden
	Thomas	Kunze	Dammweg	45	12437	Berlin
	...	...	...			

Matching Process





- Cardinalities:

1:1 `firstname` → `firstname`

1:n/n:1 `name` → `firstname`
 ↘ `lastname`

n:m
address row 1 }
address row 2 } → { street
 number
 city
 ZIP

Properties and relations of schema elements

- Notation
- Data types
- Constraints
- Key relationships
- is-a-, part-of-relations

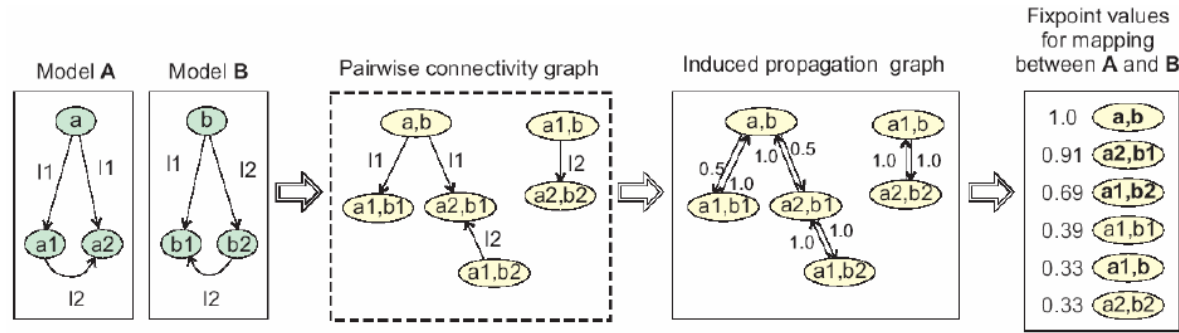
Rule-based comparison (a.k.a. Constraint-based)

- Element level
 - Data types, value ranges, key attributes
- Structure level
 - Element relations, combination of data types
 - Comparisons of element neighborhood
- Benefit: uniform language eases Matching
- Drawback: n:m Mappings → additional Matchers for safety

Schema Based Matching

Example rule based algorithm on structure level: Similarity Flooding

- Idea: Transform schemas into graphs
- Exploit the complex structure of a schema
- Given: initial similarity between schema elements (label/instance based)
- Transfer similarities to neighbors (defined by the structure)
- If neighbors of x and y are similar, (maybe) x and y also match
- Analogy: flooding a network of similarities until a balance is reached



Problems

- Other matching methods are required to get initial similarities
- Many decisions necessary
 - Which edges? Which edge weight?
 - How to normalize?
- Can be slow
- Exploitation of structural relations is an important expansion

Today: effectively used in all prototypes

Linguistic Comparisons on the element level

- Syntactic Customer $\leftrightarrow$ Person
 - Affix, N-grams (Bigrams, Trigrams)
 - EditDistance, Soundex
- Semantic Customer $\leftrightarrow$ Person
 - Synonyms, general terms
 - Foreign languages $\rightarrow$ support dictionaries

	C_Name	C_St	C_C
Customer	Emma Schmidt	Luisenstraße 90	München, 80333
	Klaus Schumann	Königstraße 7	Dresden, 01199
	...	...	...

	Firstname	Lastname	Adress			
Person			Street	Nr.	ZIP	City
	Susanne	Fröhlich	Weststraße	2	01187	Dresden
	Thomas	Kunze	Dammweg	45	12437	Berlin
	...	...	...			

Trigrams (Manning 1999)

Firstname	_ _ F	_ Fi	Fir	irs	rst	stn	tna	nam	ame	me_	e_ _
Name						_ _ N	_ Na	nam	ame	me_	e_ _

$$sim = \frac{2 \cdot |shared_trigrams|}{|trigrams(S_1)| + |trigrams(S_2)|}$$

$$sim = \frac{2 \cdot 4}{11 + 6} = 0,47$$

Schema Based Matching

Problem of schema based matching

<u>Customer</u>	C_Na		C_St		C_C	
	Firstname	Lastname	Street	Nr.	ZIP	City
	Emma	Schmidt	Luisenstraße	90	München	80333
	Klaus	Schumann	Königstraße	7	Dresden	01199
	...	...	...	...	...	...

<u>Person</u>	Person		Address			
	Firstname	Lastname	Street	Nr.	ZIP	City
	Susanne	Fröhlich	Weststraße	2	01187	Dresden
	Thomas	Kunze	Dammweg	45	12437	Berlin
	...	...	...	...	...	...



Instance Based Matching

Application

- Limited schema information
 - Restricted element identifiers
 - Semi-structured data
- Complements schema based matching

Benefit

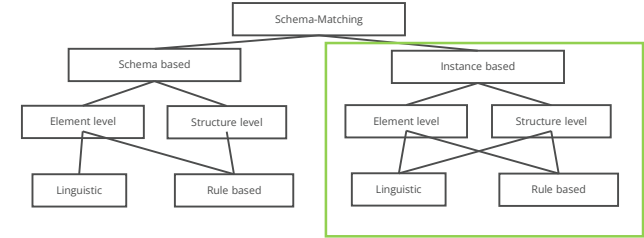
- Schema-matching without schema information

Drawback

- Needs representative instance data, data volume (sampling)

Linguistic approaches on the element level

- Frequency of words and word combinations
- Keywords, abbreviations



Instance Based Matching

Rule base comparison on the element level

- Data types, -length
- Value range, distribution, pattern
- Key attributes

Rule base comparison on the structure level

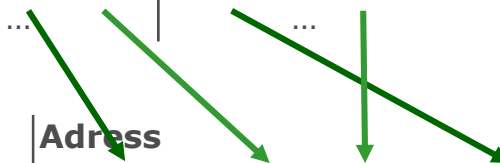
- Combination of these elements
- E.g. data types of attribute groups, tables

Instance Based Matching

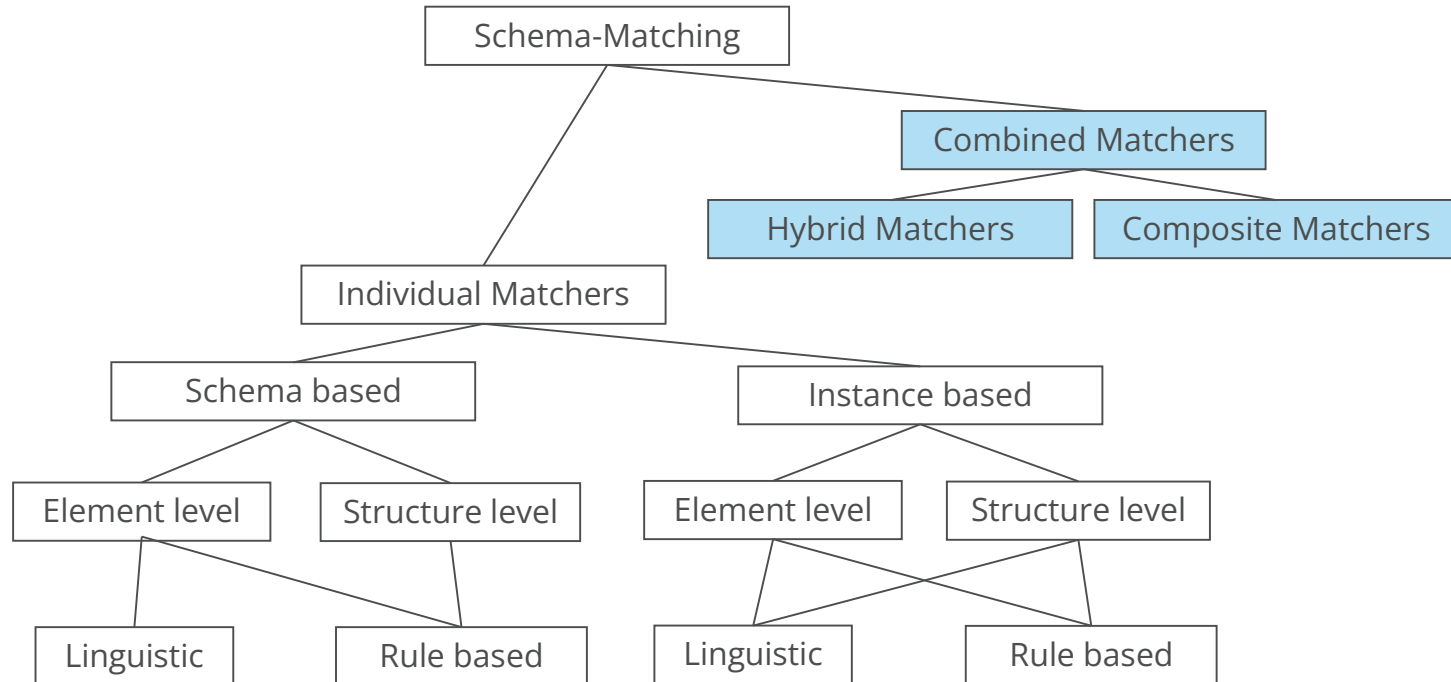
Rule based approaches of the element level

<u>Customer</u>	C_Na	C_St	C_C
	Emma Schmidt	Luisenstraße 90	München, 80333
	Klaus Schumann	Königstraße 7	Dresden, 01199
	...	...	...

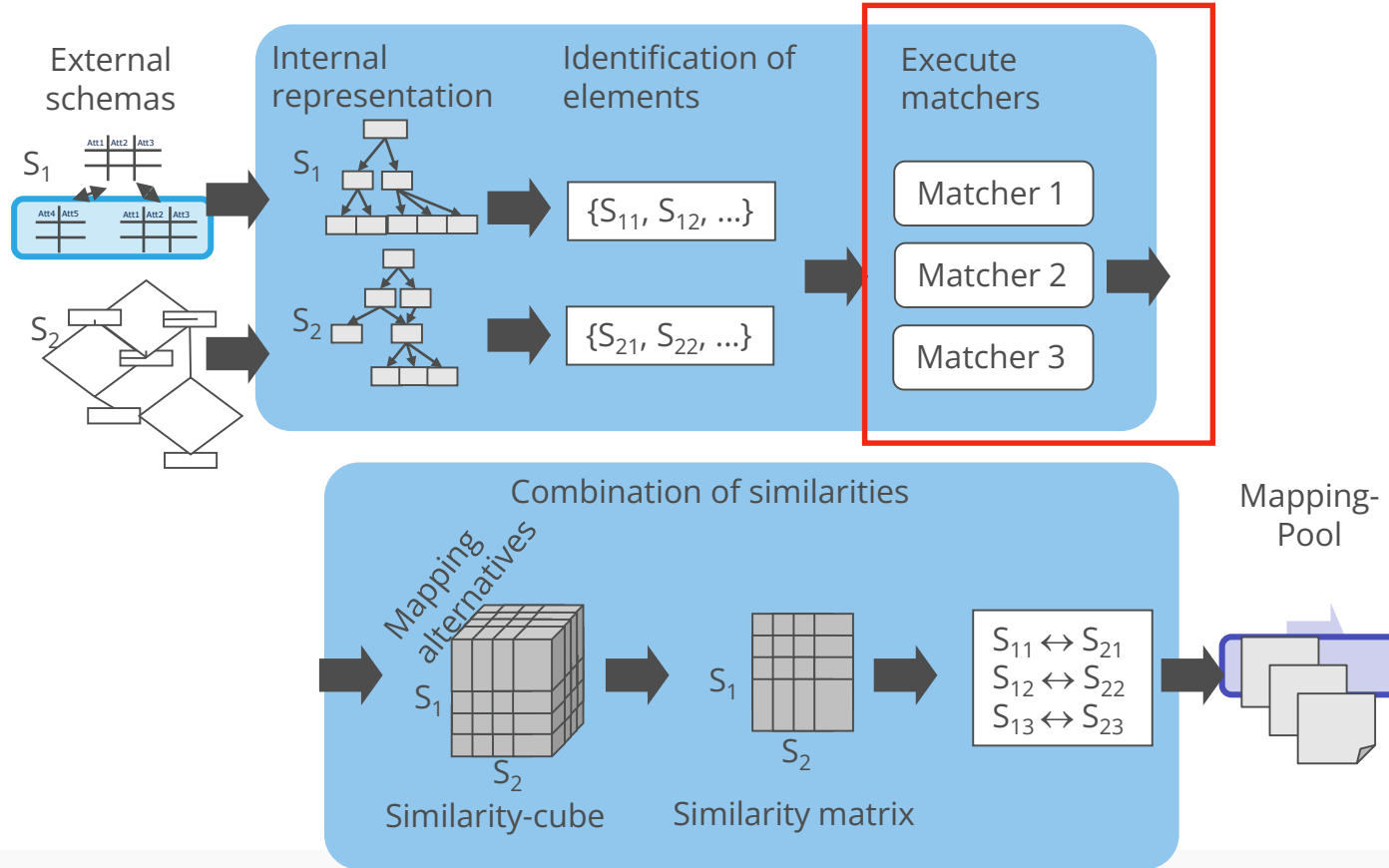
<u>Person</u>	Firstname	Lastname	Adress			
			Street	Nr.	ZIP	City
	Susanne	Fröhlich	Weststraße	2	01187	Dresden
	Thomas	Kunze	Dammweg	45	12437	Berlin
	...	...	...	...	...	...



Extended classification

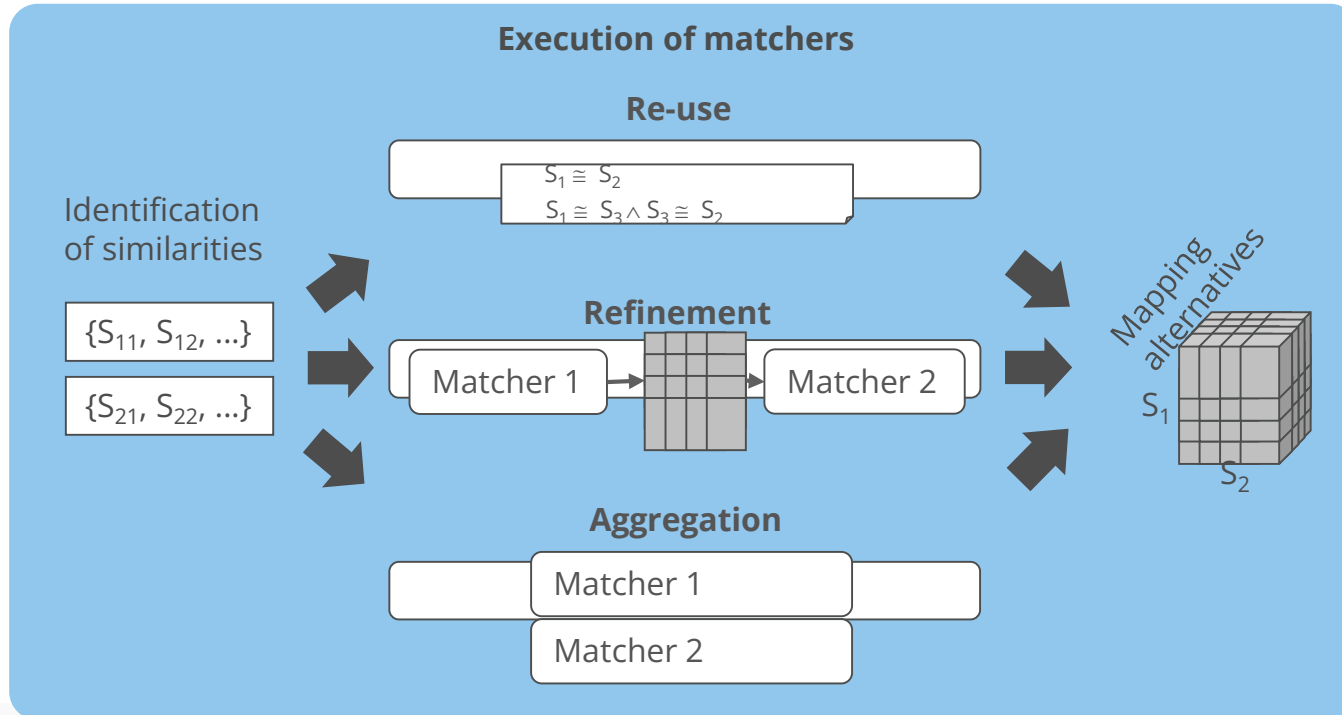
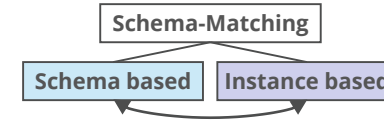


Matching Process



Combined Matching Approaches

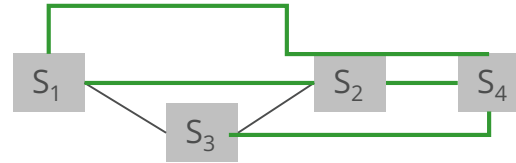
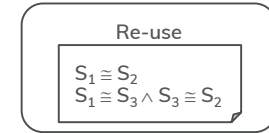
Types of matcher combination



Re-use of Mappings

Transitive Similarity

- $S_1 \cong S_3 \wedge S_3 \cong S_2 \rightarrow S_1 \cong S_2$

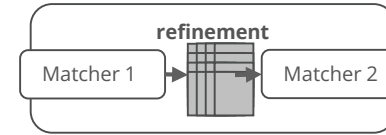


S_1	S_3	sim_{13}	S_3	S_2	sim_{32}	S_1	S_2	sim_{12}
Vorname	VName	0,8	VName	Name	0,6	Vorname	Name	0,7
Nachname	NName	0,7	NName	Name	0,6	Nachname	Name	0,65
Straße				Straße		Straße	Straße	?

Refinement of Mappings

Chaining of Matchers

- Matcher $n+1$ works only on the mapping of matcher n
- Benefit: efficient matching via “pre-selection”
- Drawback: mappings can be lost through early filtering



Context dependent matching

- Goal
 - matching of recurring schema components
- Solution:
 - 1. Evaluate context (path) of schema component → prevent wrong assignments
 - 2. Matching element of components



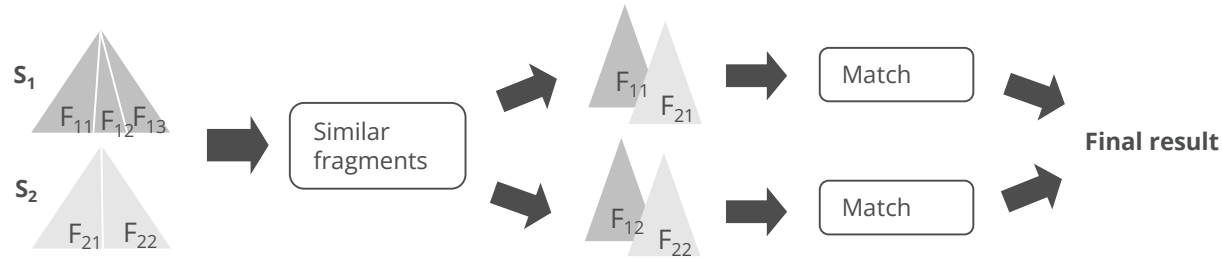
Refinement of Mappings

Problem

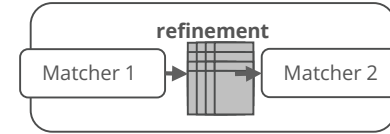
- Big schemas
- Matching of all elements is too costly $O(n \cdot m)$

Fragment based matching

- Divide & Conquer $\rightarrow$ schema reduction



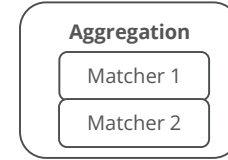
- Benefit: efficient matching
- Drawback: Fragment-spanning mappings are lost



Aggregation of Matchers

Aggregated matchers

- Independent matchers are executed in parallel



Configuration parameters

- Set of matchers
- Combination of mapping alternatives
 - (1) Aggregation
 - (2) Selection (simplest form of aggregation)

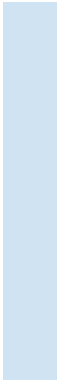
Elements S_1	Elements S_2	Matcher	Similarity
Name	Vorname	Trigram	0,53
		Synonym	0,5
Name	Nachname	Trigram	0,5
		Synonym	1,0



Elements S_1	Elements S_2	Similarity
Name	Vorname	0,51
Name	Nachname	0,75



Elements S_1	Elements S_2	Similarity
Name	Nachname	0,75



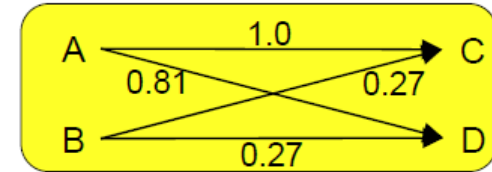
Global Schema Matching

So far: each attribute matching is decided individually

- Actually, complete schemas should be mapped

Generally, optimal match for each attribute is not possible

- Multiple equal targets: should B be mapped to C or D?
- Best source/target for several attributes (A for C and D)



→ Global Matching

- Find a set of 1:1 relations between attributes that is globally optimal
- Classical formulation: find maximal **bipartite matching**
- Alternative: **Stable Marriage**

Example: Stable Marriage

Stable Marriage

Given

- n women (attributes in schema A) and m men (attributes in schema B)
- **Monogamy:** one woman can only be married to one man and vice versa (only 1:1 matches)
- **Preferences:** each woman has a ranked list of men and vice versa, typically symmetric

Wanted: pairing (global matching), so that it never holds that

- w_1 marries m_1 , w_2 marries m_2 ,
- but w_1 prefers m_2 and m_2 prefers w_1 (marriage would be unstable!)

Example stable marriage

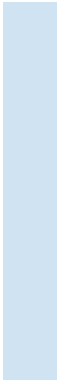
Example

Men(1-4)	Women(A-D)
1: B, D, A, C	A: 2, 1, 4, 3
2: C, A, D, B	B: 4, 3, 1, 2
3: B, C, A, D	C: 1, 4, 3, 2
4: D, A, C, B	D: 2, 1, 4, 3

Possible steps

- 1 proposes to B, she agrees: (1, B)
- 2 proposes to C, she agrees : (1, B) (2, C)
- 3 proposes to B, she agrees & leaves 1: (2, C) (3, B)
- 1 proposes to D, she agrees : (1, D) (2, C) (3, B)
- 4 proposes to D, she declines : (1, D) (2, C) (3, B)
- 4 proposes to A, she agrees : (1, D) (2, C) (3, B) (4, A)

Four stable
pairs!



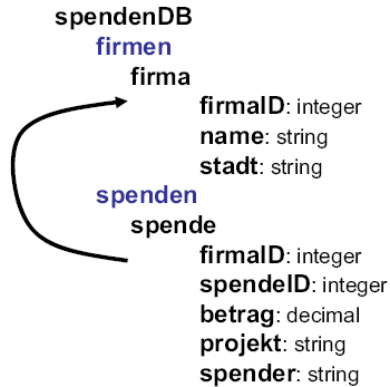
Mapping Interpretation

Motivation

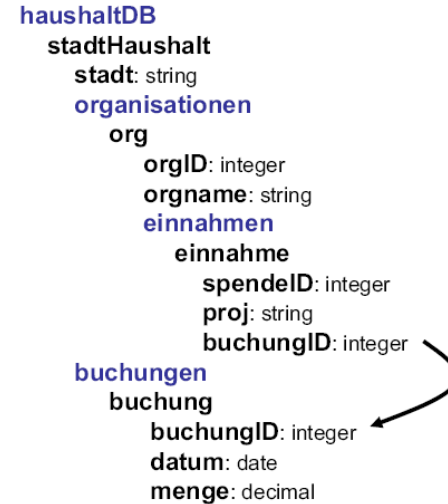
- Given: two schemas
 - Created independently
 - Relational
 - Nested
 - Integrity constraints (PK / FK)
 - Partly model different data
- Given: set of correspondences between schemas
- Wanted: transformation queries that keep the original semantic, consider target schema integrity constraints, consider all correspondences if possible

Mapping Interpretation

Example



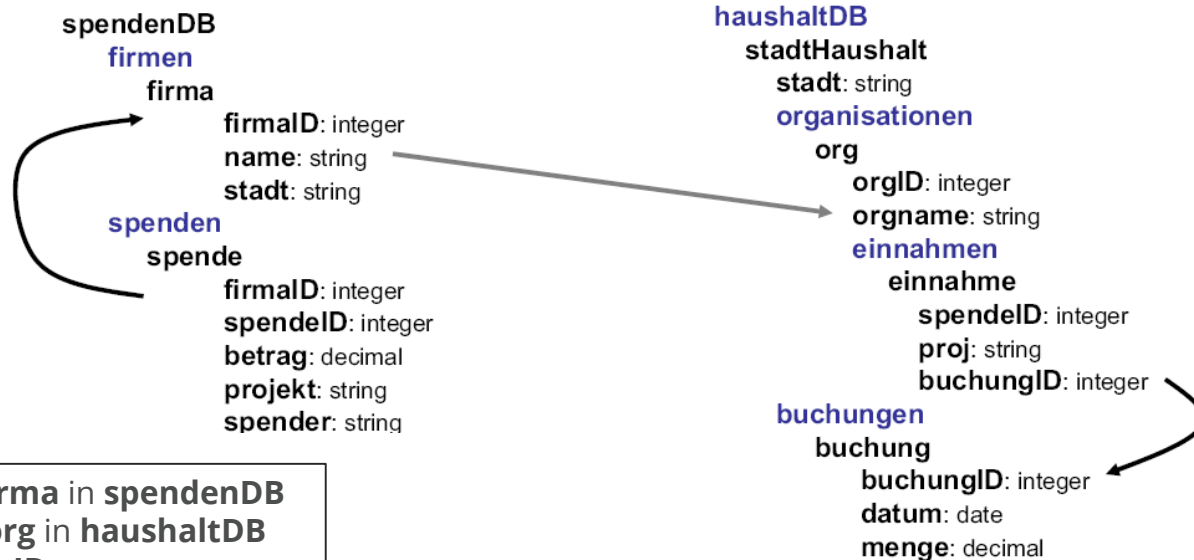
Source



Target

Interpretation

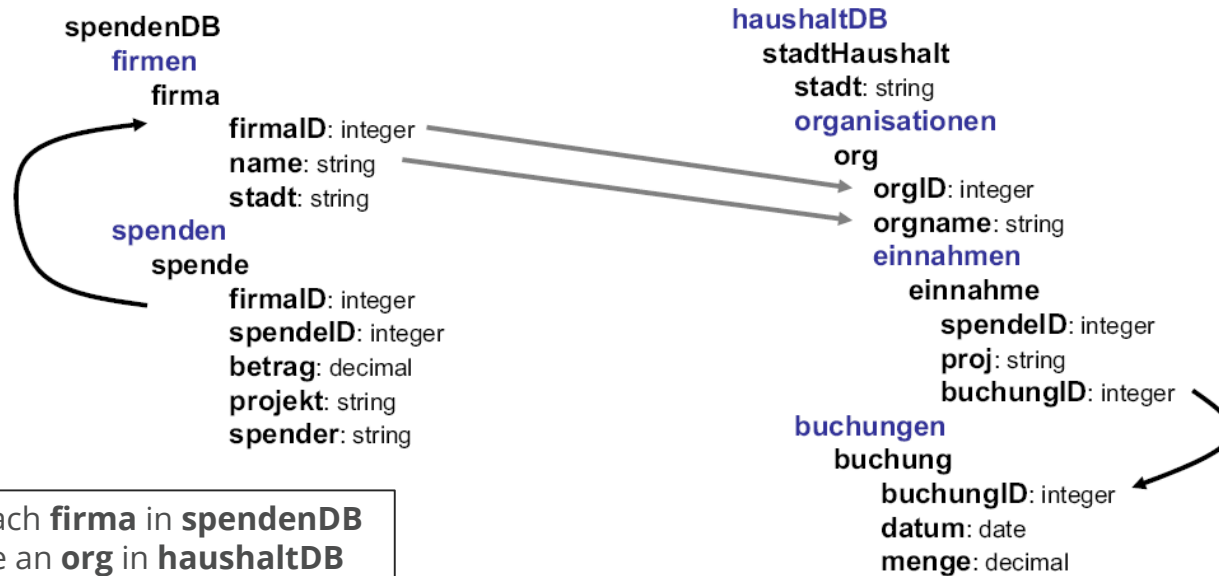
- Mapping of name



- For each **firma** in **spendenDB** create an **org** in **haushaltDB**
- „invent“ **orgID**
- „invent“ **city**

Alternative Interpretation

- Mapping of ID and name



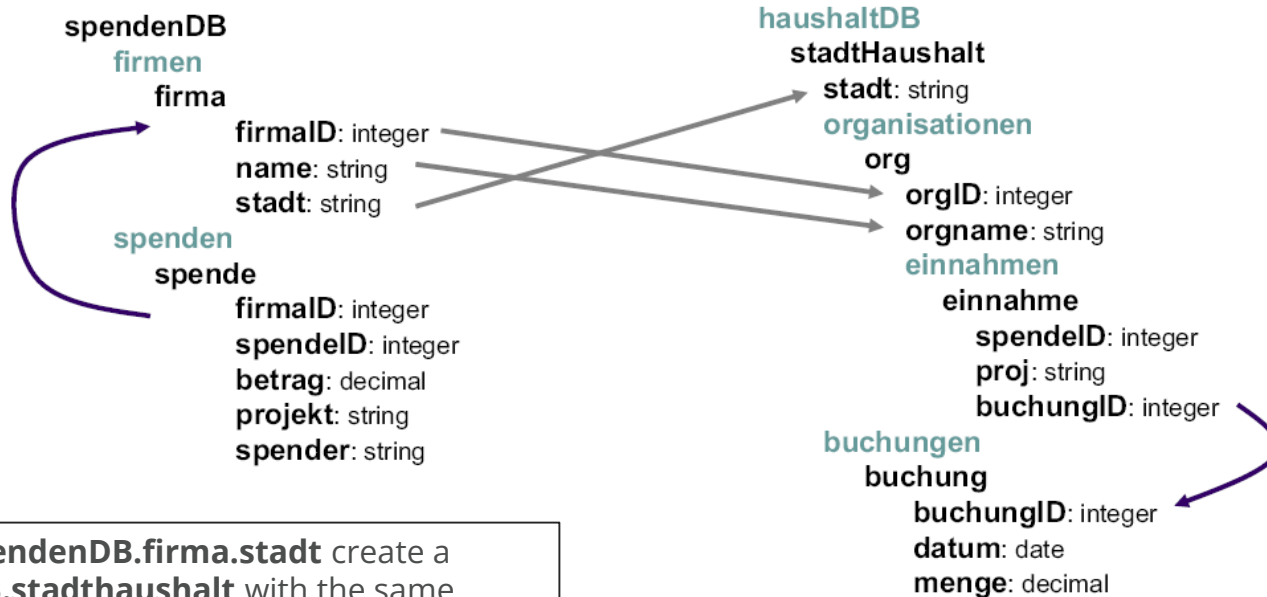
- For each **firma** in **spendenDB** create an **org** in **haushaltDB**
- „invent“ city

“Inventing” Values

Two reasons: non-null and identity

- Non-NULL values
 - Invented value does not matter: „unknown“ or „null“ (or „Dresden“)
- ID values: skolem function
 - Input: n Values
 - Output: distinct values w.r.t. input
 - E.g. concatenation of all input values as string

Mapping Interpretation



- For each **spendenDB.firma.stadt** create a **haushaltDB.stadthaushalt** with the same name
- Group each **firma** under its respective **stadtHaushalt**

Algorithm for Schema Mapping

Steps

1. Identify intra-schema associations
2. Identify inter-schema logical mappings
3. Generate queries

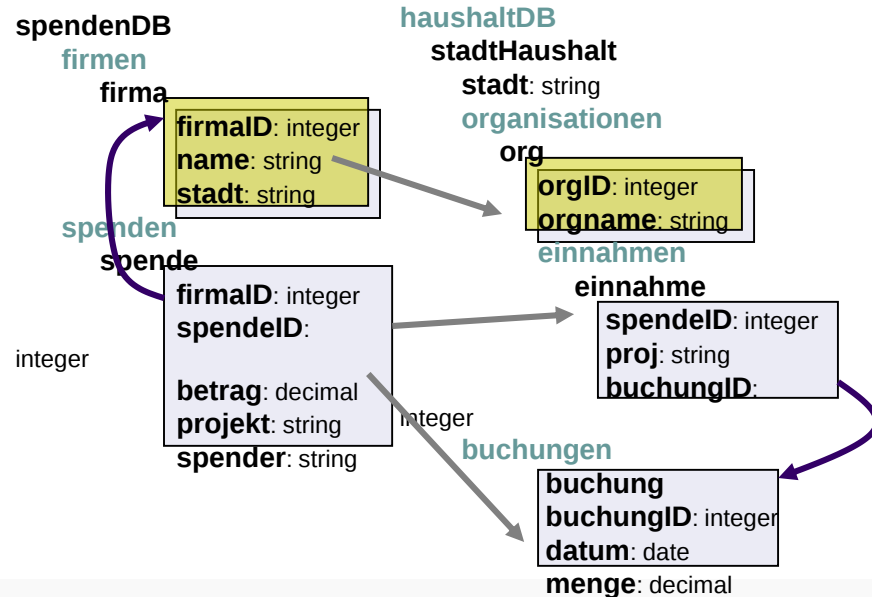
Different interpretations:



Companies without donations

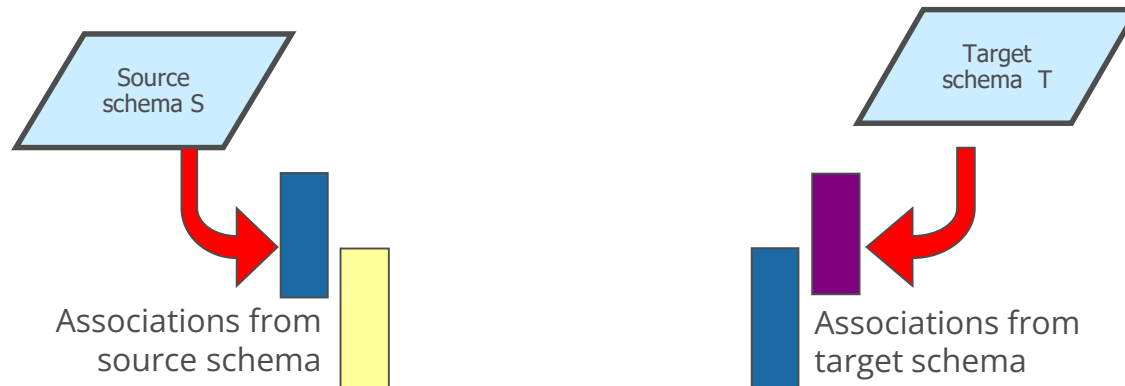


Companies with donations



Step 1: Identify intra-schema associations

- Associations between schema elements
- Relational Views contain maximal groups of associated elements (same relation, nesting, PK-FK-relationship)
- Each View represents its own “category” of data from the data source
- Independent from mapping (but limited to “mapped” elements)

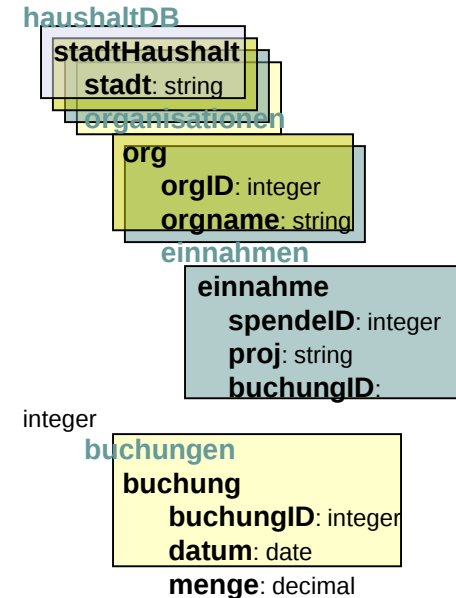


Intra-schema Associations

Step 1: Identify intra-schema associations

Start

- Starting point: all primary paths
- Possible associations in the schema without integrity constraints
- Relational schemas
 - Each relation corresponds to a primary path
- Nested schemas
 - Attribute of one level and nested levels

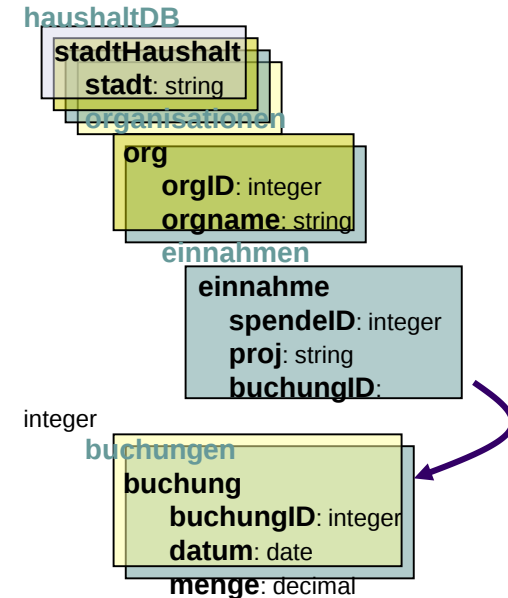


Intra-schema Associations

Step 1: Identify intra-schema associations

Consider Primary Keys / Foreign Keys (ICs)

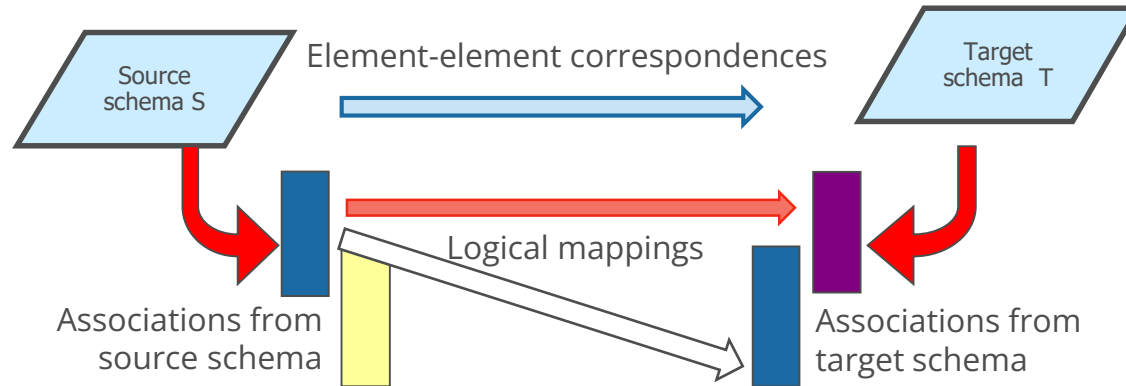
- Creation of “logical relations”
- Extend all primary paths via „Pursuing” of ICs



Identify Logical Mappings

Step 2: Identify Inter-Schema-logical mappings

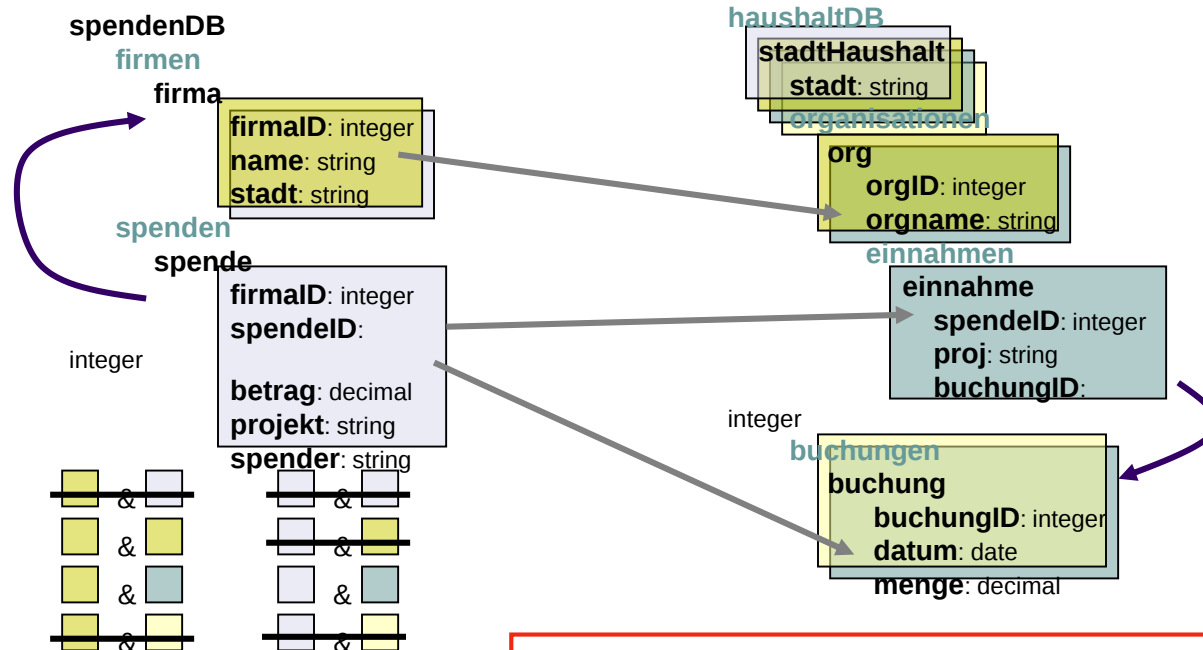
- Find logical mappings between source and target schema
- Consider all combinations of associations of source / target schemas



- User/ domain expert input → semi-automatic process
 - User selects logical mapping (interpretations)
 - User “paints” correspondences

Identify Logical Mappings

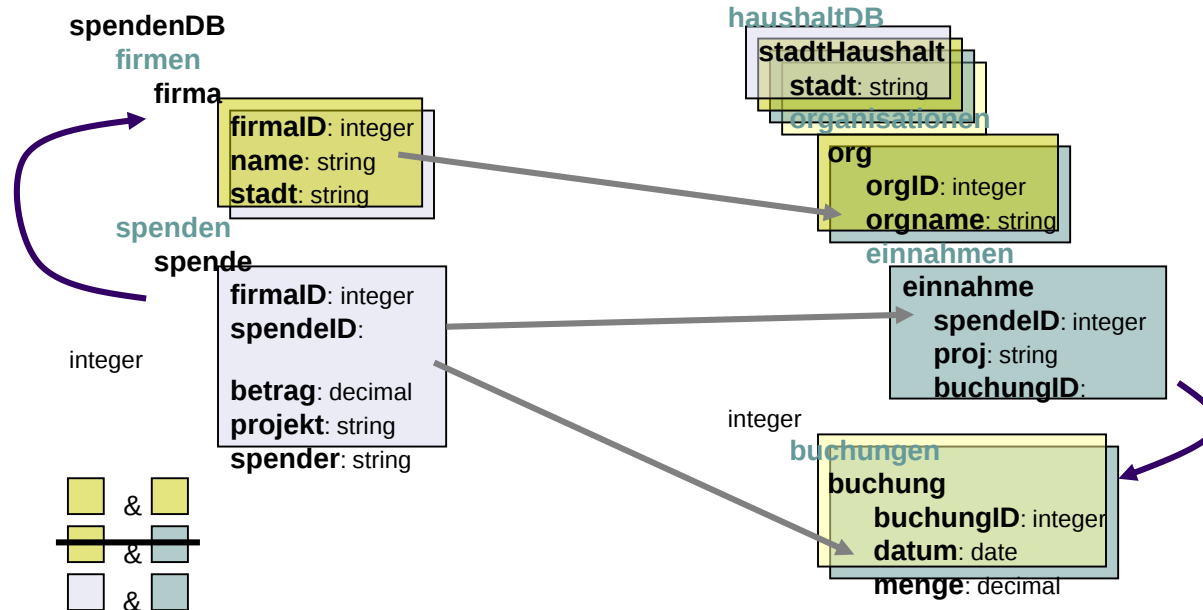
Step 2: Identify Inter-Schema-logical mappings



Do all outgoing correspondences find a target in the combination?

Identify Logical Mappings

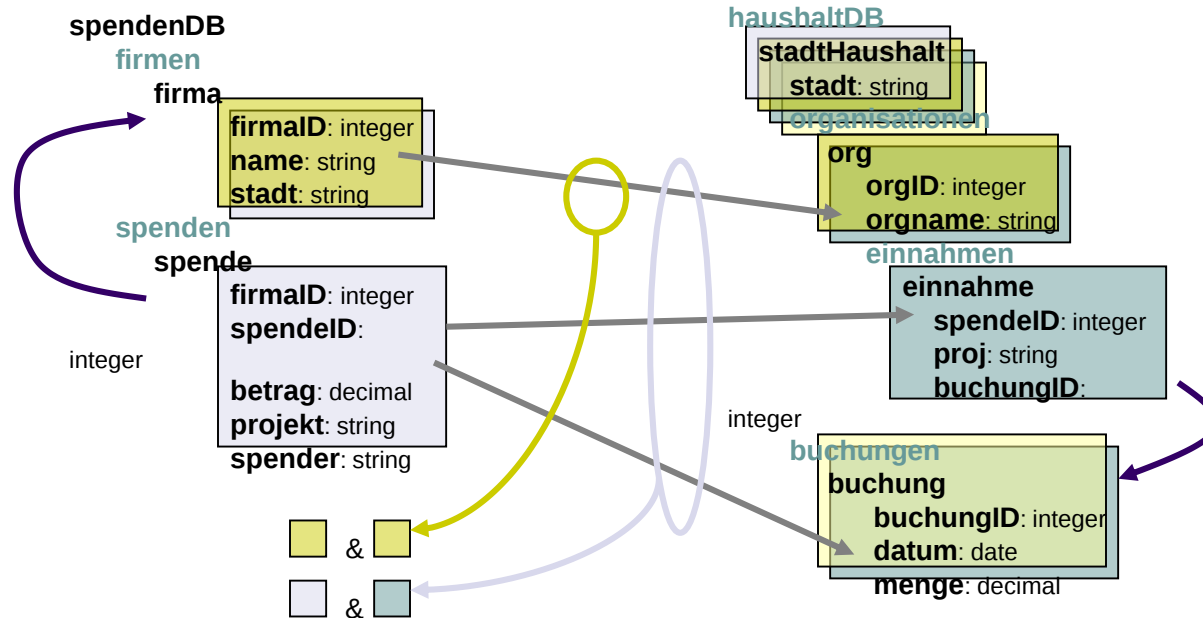
Step 2: Identify Inter-Schema-logical mappings



Do all incoming correspondences originate from the combination?

Identify Logical Mappings

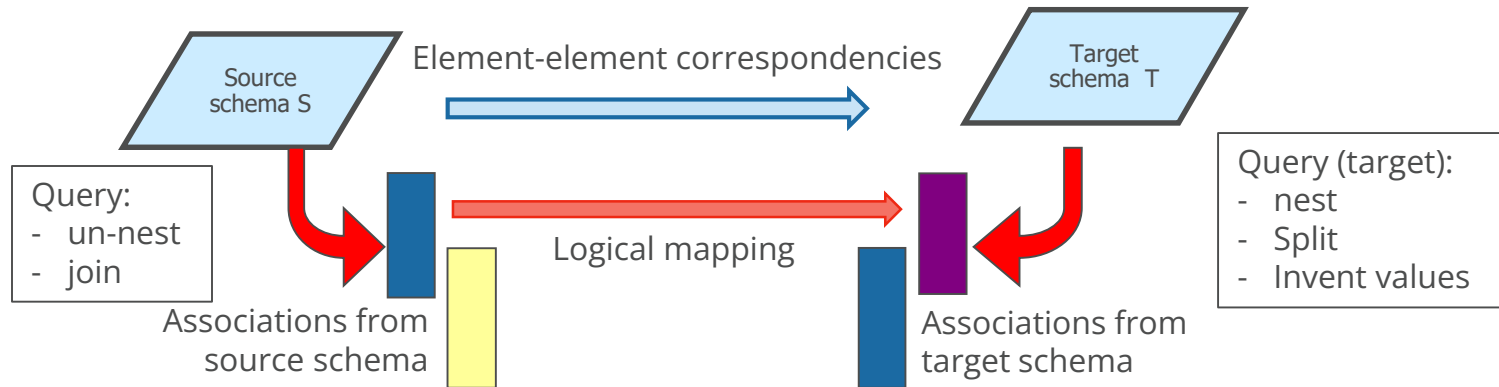
Step 2: Identify Inter-Schema-logical mappings



Step 3: Generate queries

Create a query for each selected logical mapping

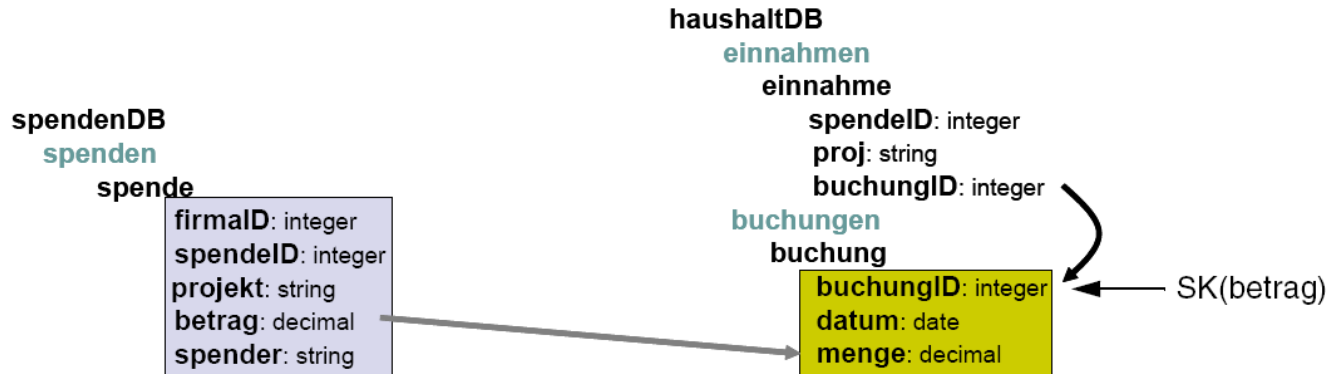
- Select and join source data
- Generate target data



Step 3: Example relation: buchungen

Value for buchungID is not set

- Either: dont care
- Or: Not-null: needs default value, e.g. „null“
- Or: ID: requires creation of unique values
- Skolem function, based on all values of the mapping of this logical relation



Generate Queries - Grouping

All attributes get a value, but

- Possible loss of associations
- Possible creation of new (incorrect) associations
- Grouping necessary

Method

- Virtual ID creation using skolem function based on all values hierarchically above the current relation
- Example: each tuple **haushaltDB** gets ID $Sk(\text{land}, \text{stadt})$
- Each tuple from **firmen** with same **stadt** and **land** gets same ID is nested below the same element

